

Figure 1.





# Step 101

The **Sender** creates the message content (MailContent) and selects a random encryption key (SymmetricKey). Both MailContent and SymmetricKey should be kept by the **Sender** in order to verify the validity of the certified receipt later.

The Sender sends to the Recipient the certified mail defined as:

CertifiedMail = PKE(RemailerPublicKey, CertMailHeader) + CertMailBody where:

#### **Step 102**

CertMailHeader = MessageID + SymmetricKey;

CertMailBody = HASH(SymmetricKey) + SKE(SymmetricKey, MailContent); MessageID = HASH(CertMailBody);

## Step 103

After receiving CertifiedMail, the *Recipient* sends a receipt to the *Remailer*:

ReceiptSentToRemailer = PKE(RemailerPublicKey, CertMailHeader) +

HASH(SymmetricKey) + SignedReceipt

Where: SignedReceipt = SIGNED(RecipientPrivateKey, MessageID2) and MessageID2 is the message ID the *Recipient* computed from the received message according to: MessageID2 = HASH(CertMailBody);

The Remailer processes ReceiptSentToRemailer as the following:

- a) Decrypts PKE(RemailerPublicKey, CertMailHeader) to obtain SymmetricKey and MessageID from CertMailHeader.
- b) Verifies SignedReceipt using the public key of the *Recipient*.

## Step 104

- c) Verifies that MessageID obtained from CertMailHeader is exactly the same as MessageID2 in SignedReceipt.
- d) Verifies that HASH(SymmetricKey) in the ReceiptSentToRemailer agrees with the hash computed from SymmetricKey in CertMailHeader.
- e) If all the verifications succeed, send the SignedReceipt to the Sender.
- f) If sending receipt to the **Sender** succeeds, send the SymmetricKey to the **Recipient**.

#### **Step 105**

The *Recipient* decrypts SKE(SymmetricKey, MailContent) using the SymmetricKey received from the *Remailer* to obtain MailContent.

After receiving the SignedReceipt, the **Sender** is able to prove that the recipient has received the exact MailContent by demonstrating:

#### **Step 106**

- a) The *Recipient's* signature signed SignedReceipt can be verified using *Recipient's* public key.
- b) The MessageID2 in the SignedReceipt agrees with the hash of CertMailBody reconstructed from SymmetricKey and MailContent the *Sender* has kept.



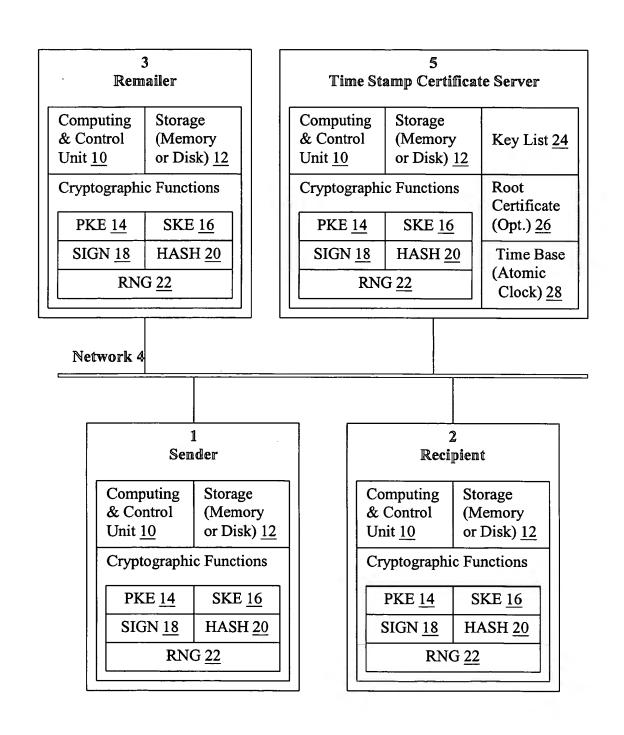


Figure 3.





Step 401

The **Sender** creates the message content (MailContent) and selects a random encryption key (SymmetricKey).

Step 402

The **Sender** constructs CertMailBody and computes MessageID CertMailBody = HASH(SymmetricKey) + SKE(SymmetricKey, MailContent); MessageID = HASH(CertMailBody);

Then, the **Sender** sends MessageID, SenderAddress, RecipientAddress, and RemailerAddress to the **TSC Server** to retrieve a TSC for the sending time.

The TSC Server issues a TSC for the sending time:

SendTSC = SIGNED(TSCServerPrivateKey, MessageID + SendTime + SenderInfo + RecipientInfo + RemailerInfo + RootCertificate):

Step 403 Senderinto + Recipientinto + Remailerinto + Rewalterinto + Rewalterinto + Remailerinto + Remailerinto

SenderInfo = SenderAddress + SenderPublicKey

RecipientInfo = RecipientAddress + RecipientPublicKey

RemailerInfo = RemalerAddress + RemailerPublicKey

The Sender verifies SendTSC, constructs the signed certified mail header:

SignedCertMailHeader = SIGNED(SenderPrivateKey, SendTime + MessageID + SymmetricKey)

and then sends the Recipient the certified mail defined as:

CertifiedMail = PKE(RemailerPublicKey, SignedCertMailHeader) +

+ PKE(RecipientPublicKey, SignedCertMailBody);

where:

Step 404

SignedCertMailBody = SIGNED(SenderPrivateKey, CertMailBody + SendTSC).

The Sender also keeps a "carbon copy" of the certified message:

CarbonCopy=PKE(SenderPublicKey, SignedCertMailHeader) +

+ PKE(SenderPublicKey, SignedCertMailBody);

**Step 405** 

After receiving CertifiedMail, the *Recipient* decrypts the second part to obtain SignedCertMailBody, verifies it, computes MessageID2=HASH(CertMailBody), and then sends MessageID2, RecipientAddress, SenderAddress, and RemailerAddress to *TSC Server* to retrieve a TSC for the receiving time.

Continued to Figure 4b

Figure 4a

### Continued from Figure 4a

**Step 406** 

The *TSC Server* issues a TSC for the receiving time:

ReceiveTSC = SIGNED(TSCServerPrivateKey, MessageID2 +

ReceiveTime + RecipientInfo + SenderInfo + RemailerInfo + RootCertificate):

Step 407

The *Recipient* verifies the ReceiveTSC and sends a receipt to the *Remailer*:

ReceiptSentToRemailer = PKE(RemailerPublicKey, SignedCertMailHeader) +

PKE(RemailerPublicKey, HASH(SymmetricKey) + ReturnSessionKey +

SignedReceipt), where:

SignedReceipt = SIGNED(RecipientPrivateKey, SendTSC + ReceiveTSC)

Step 408

The *Remailer* decrypts ReceiptSentToRemailer to obtain SignedCertMailHeader, HASH(SymmetricKey), and SignedReceipt. Then, the *Remailer* conducts a series of verification steps to ensure that the SignedCertMailHeader, SignedReceipt, SendTSC, ReceiveTSC are all valid and the data contained in them are all consistent. If all the verifications succeed, the *Remailer* sends the *Sender* CertifedReceipt = PKE(SenderPublicKey, SignedReceipt) and sends SKE(ReturnSessionKey, SymmetricKey) to the *Recipient*.

Step 409

The *Recipient* decrypts SKE(ReturnSessionKey, SymmetricKey) received from the *Remailer* to recover SymmetricKey and then use it to decrypt SKE(SymmetricKey, MailContent) to obtain MailContent.

Step 103

After receiving the CertifedReceipt, the **Sender** is able to prove that the MailContent existed at SendTime and is delivered to the recipient at ReceiveTime by demonstrating:

**Step 410** 

- a) The *Recipient's* signature in SignedReceipt can be verified using RecipientPublicKey in ReceiveTSC.
- b) The MessageID or MessageID2, in SignedReceipt, SendTSC, ReceiveTSC, all agrees with the hash of the CertMailBody recovered from the CarbonCopy kept by the *Sender* during Step 404 above.
- c) SenderInfo, RecipientInfo, RemailerInfo in both SendTSC and ReceiveTSC are all consistent.
- d) The signatures in SendTSC and ReceiveTSC can be verified using the *TSC*Server's public key in the RootCertificate, and the RootCertificate can be verified using the root public keys.
- e) SendTSC in CarbonCopy is the same as the one in the SignedReceipt.

Figure 4b